ISO TC184/SC4/WG12 N511

First Edition: 2000-1-31

[File: Pc/Word]

ISO 10303 White Paper

Parametric Representation and Exchange: Parametric assembly constraints in explicit parametric model representation

ABSTRACT:

The series of International Standards to be produced by the project ISO 10303 WG12 / Parametrics deal with the representation and exchange of information associated with parametrization and constraints as applied to product model shape. It defines methodologies for vendor and implementation independent representation and exchange of parametric models.

The present document is a White Paper concerned with the provision of assembly constraints which is an essential requirement for the capture and exchange of parametric models. The document presents necessary constraint types with related shape elements. A schema dealing with such capabilities in vendor independent specification will form part of the ISO 10303-Part108standard.

KEYWORDS: assembly feature, assembly feature association, parametric assembly constraint, distance constraint, angle constraint, parallel constraint, perpendicular constraint, coincident constraint, cocentric constraint, tangent constraint, fixed component constraint

COMMENTS TO READER:

This document is intended as a discussion document, and informed comment is invited, either via the SC4 Parametrics exploder or directly to the editor.

Project Leader: Michael J. Pratt
Address:
NIST
Building 220, Room A127
Gaithersburg, MD 20899-0001
USA
Telephone: +1 (301) 975-3951
Telefax: +1 (301) 975-4482
Electronic mail: pratt@nist.gov

Document Editor: Akihiko Ohtaka
Address:
Nihon Unisys, Ltd.
1-1-1 Toyosu, Koto-ku
Tokyo 135-8560 Japan
Telephone:+81 3 5546 4784
Telefax: +81 3 5546 7810
Electronic mail: akihiko.ohtaka@unisys.co.jp

Parametric Assembly Constraints in Explicit Parametric Model Representation

First ed.: 1st Feb. 2000

Nihon Unisys, Ltd.
Akihiko Ohtaka

Contents

page

7 Schema parametric_assembly_constraint

7.1 Introduction

The following EXPRESS declaration begins the **parametric_assembly_constraint_schema** and identifies the necessary external references.

EXPRESS specification:

```
*)
SCHEMA parametric_assembly_constraints_schema;
   REFERENCE FROM explicit_geometric_constraint_schema
       (explicit_geometric_constraint);
   REFERENCE FROM assembly_model_schema
       (assembly_feature,
       assembly_feature_association);
   REFERENCE FROM measure_schema
       (length_measure,
       plane_angle_measure);
   REFERENCE FROM geometry_schema;
(*
```

7.2 Fundamental concepts and assumptions

This schema provides the resources for the specification of explicit geometric constraints between component parts of an assembly product. Significant difference of explicit geometric constraints defined in chapter-6 and constraints defined in current schema is that location and size of individual geometric element in a component part is variable within the range of constraints in the former, but geometric relationship between two **rigid** component parts which corresponds to transformation matrix from one to the other is variable within the range of constraints in the latter.

Component parts are instances of **product_definitional_shape** and geometric elements which appear in constraint definitions are instances of subtypes of **geometric_representation_item** which belong to target component parts. The particular set of valid subtypes differ from one constraint type to another. In most cases, they are subset of simple geometric types such as point, line, circle, plane, cylindrical surface, conical surface and spherical surface. None of these geometries are bounded in this schema. Some constraints have additional attributes to specify desired solution when two or more solutions are possible. These are attributes to specify if directions are identical or opposite, to specify in which side desired solution exists, and to specify near point. Care should also be taken if a constraint is measured to the centre of a geometry or to its boundary. A parametric assembly constraint is applied in an assembly model composed of geometrical (and topological) component parts. It assets a relationship between two such component parts present in the model. The fundamental concepts and assumptions stated in Clause 5.2 of the **explicit_geometric_constraint_schema** also apply in this schema.

7.2.1 Interpretation of predefined constraints in this schema

All the constraints defined in this schema are of predefined class. They are specified in descriptive geometric terms, the assumption being that shape modeling systems provide implementations of corresponding constraints and will have a precise understanding of their semantics. Thus when a model with parametric assembly constraints is transferred into a receiving system, the transmitted descriptive constraints can be interpreted by the system in whatever form is most appropriate for its internal functionality. In particular, the descriptive form can be reformulated in the explicit mathematical manner best suited to the nature of the receiving system's constraint solving.

7.2.2 Categorization of parametric assembly constraints

There are two categories of parametric assembly constraints. One is dimensional constraint, the other is logical constraint. **Distance_constraint** and **angle_constraint** are the former type constraints, and all the other constraints are the latter type constraints. Although distance and angle constraints can take a value of zero, it is recommended to use appropriate logical constraint in that situation in order to explicitly declare the constraint condition between two component parts irrespective of the difference of numerical accuracy treatment of the sender's system and the receiver's system . A zero value dimension is useful in some special situation such as forcing two **axial_geometry** type entities to intersect by specifying zero distance between two non-planar lines.

7.2.3 Dimensionality of parametric assembly constraints

The constraints defined in this schema are valid in both two and three dimensions. Frequently used case in commercial systems is three dimensional case where detailed shape of component part is represented by a solid model, a surface model or their mixture. But, the schema could also be used

in earlier stage of design where outline of component parts are simply represented by two dimensional model. If target geometries are contained to lie in a plane, any constraint will be interpreted in the same way as described in Chapter-6.

7.2.4 Constraints to curved geometry

Circles, spherical surfaces, cylindrical surfaces and conical surfaces are defined as curved elements(see 7.3.4 below). A constraint to a curved element may have more than one interpretation. For some types of constraints to circles, spherical surfaces, and cylindrical surfaces, this schema allows constraints measured to;

1) The geometry centre

2) The nearest geometry boundary, or

3) The geometry boundary which is close to specified ‘near point’

For distance constraint, tangent constraint, and for coincident constraint to circles, spherical surfaces or cylindrical surfaces, near point(s) can be specified for focusing on the desired solution.

7.3 parametric_assembly_constraint type definitions

This clause contains the TYPE definitions of the **parametric_assembly_constraint_schema.**

I checked specifications of constraints and target shape of 3D DCM, UG, Solid Works, and CADCEUS. Constraint types and target geometries specified below mostly satisfy specifications of these systems. But, only 3D DCM allows swept surface, general parametric curve, and general parametric surface as target geometries in some types of constraints. Intended applications I hear are cam design and ship hull design. Since these applications are some rare examples and inclusion of above type geometries makes the specification complicated, I intentionally did not include these geometries.

7.3.1 directed_geometry

The **directed_geometry** type comprises lines, circles, planes and cylindrical surfaces. As for the directions, the line direction for a line, the axis direction for a circle or for a cylindrical surface and the normal direction for a plane are implied in this type. A parallel constraint, for instance, between a cylindrical surface and a line will act as if it were applied between the axis of cylinder and the line. Care should be taken, however, that parallel constraints between a line and a plane or a circle implies that the line is perpendicular to plane direction or circle direction.

EXPRESS specification:

```
*)
TYPE directed_geometry = SELECT
   (line,
   circle,
   plane,
   cylindrical_surface);
END_TYPE;
(*
```

7.3.2 axial_geometry

The **axial_geometry** type comprises lines, circles, and cylindrical surfaces. These have an associated axis. An axis not only has a direction, but also has a location. The axis of a line is the line itself, The axes of a circle and a cylindrical surface are the conventional one which appears in these entity definitions.

EXPRESS specification:

```
*)
TYPE axial_geometry = SELECT
    (line,
    circle,
    cylindrical_surface);
END_TYPE;
(*
```

7.3.3 all_elementary_geometry

The **all_elementary_geometry** type comprises cartesian points, lines, conics and elementary surfaces. This type is prepared for use in some constraint which is applicable to these geometry types.

EXPRESS specification:

```
*)
TYPE all_elementary_geometry = SELECT
    (cartesian_point,
    line,
    conic,
    elementary_surface);
END_TYPE;
```

(*

7.3.4 curved_geometry

The **curved_geometry** type comprises circles, spheres and cylindrical surfaces. Several important kinds of constraints involve only elements of these kinds.

EXPRESS specification:

```
*)
TYPE curved_geometry = SELECT
   (circle,
   spherical_surface,
   cylindrical_surface);
END_TYPE;
(*
```

7.3.5 elementary_curve_or_surface_geometry

The **elementary_curve_or_surface_geometry** type comprises lines, conics and elementary surfaces. This type is prepared for use in some constraint which is applicable to these geometry types.

EXPRESS specification:

```
*)
TYPE elementary_curve_or_surface_geometry = SELECT
   (line,
   conic,
   elementary_surface);
END_TYPE;
(*
```

7.4 parametric_assembly_constraint entity definitions

This clause contains ENTITY definitions of the **parametric_assembly_constraint_schema.**

7.4.1 parametric_assembly_constraint

This entity is the supertype of all parametric assembly constraints, and a subtype of

assemby_feature_association which is defined in Part1xx: Assembly Model of Products.
All the constraint treated in this schema are undirected constraint.

Parametric_assembly_constraint is defined between two geometric elements existing in different component parts or difference instances of the same component part. Those geometric entities are identified via **assembly_feature** defined in Part1xx. Though **assembly_feature.shape** has a SELECT type: **shape_aspect** or **geometric_representation_item**, this schema uses **geometric_representation_item**. Effective geometric element types depend on constraint type as will be clarified in each constraint description.

EXPRESS specification:

```
*)
ENTITY parametric_assembly_constraint
   ABSTRACT SUPERTYPE OF(ONEOF(distance_constraint,
                              angle_constraint,
                              parallel_constraint,
                              perpendicular_constraint,
                              coincident_constraint,
                              cocentric_constraint,
                              tangent_constraint))
   SUBTYPE OF (assembly_feature_association);
DERIVE
   constrained_element : assembly_feature :=
   self\assembly_feature_association.related_assembly_feature;
   reference_element : assembly_feature :=
   self\assembly_feature_association.relating_assembly_feature;
WHERE
   WR1: constrained_element:<>:reference_element;
   WR2: (‘GEOMETRIC_REPRESENTATION_ITEM’ IN TYPEOF(self.constrained_element));
   WR3: NOT(‘SHAPE_ASPECT’ IN TYPEOF(self.constrained_element));
   WR4: (‘GEOMETRIC_REPRESENTATION_ITEM’IN TYPEOF(self.reference_element));
   WR5: NOT(‘SHAPE_ASPECT’IN TYPEOF(self.reference_element));
END_ENTITY;
(*
```

Attribute definitions:

constrained_element: Geometric element which is constrained with respect to the reference element with specified constraint. This corresponds to **related_assembly_feature** attribute of the **assembly_feature_association**.

reference_element: Geometric element with which the constrained element is constrained with specified constraint. This corresponds to **relating_assembly_feature** attribute of the **assembly_feature_association**.

Formal propositions:

WR1: **constrained_element** and **reference_element** should be different instance.

WR2: **geometric_representation_item** should be selected from SELECT type of **assembly_feature.shape** for **constrained_element**.

WR3: **shape_aspect** should not be selected from SELECT type of **assembly_feature.shape** for **constrained_element**.

WR4: **geometric_representation_item** should be selected from SELECT type of **assembly_feature.shape** for **reference_element**.

WR5: **shape_aspect** should not be selected from SELECT type of **assembly_feature.shape** for **reference_element**.

7.4.2 distance_constraint

This constraint specifies that the distance in three dimensions of the constrained element in one component part from the reference element in another component part is constrained with specified value. This constraint can be applied between any pair of **all_elementary_geometry** type

geometric entities. If the geometries do not intersect and no near point is specified, then the distance is the minimum distance between them. If near point(s) is specified in this case, the distance becomes closest one among possible distances. If they do intersect, it is the smallest of the possible separation distances. In case both constrained element and reference element are planes, a BOOLEAN attribute is required to distinguish if plane normal directions are identical or opposite.

In case at least one of constrained element or reference element is a plane, another BOOLEAN attribute is required to distinguish if the element other than a plane exits in the front side of the plane or not. If this attribute is TRUE in point vs. plane case, the point exists in the front side of the plane. If both elements are planes, TRUE specifies that the constrained plane exists in the front side of the reference plane. If sense attribute is FALSE in the plane vs. plane situation, then if side attribute is TRUE, both planes exist in the front side each other, else rear side each other.

If a circle or a cylindrical surface or a spherical surface appear in one or both of constrained element and reference element, the distance is measured to its axis unless ‘centre_check’ attribute is specified FALSE.

EXPRESS specification:

```
*)
ENTITY distance_constraint
  SUBTYPE OF (parametric_assembly_constraint);
    distance: length_measure;
    sense_check : OPTIONAL BOOLEAN;
    side_check: OPTIONAL BOOLEAN;
    centre_check: OPTIONAL BOOLEAN;
    near_points: OPTIONAL SET[1:2] OF CARTESIAN_POINT;
WHERE
  WR1 : distance>=0;
  WR2: (‘ALL_ELEMENTARY_GEOMETRY’ IN
  TYPEOF(self\parametric_assembly_constraint.constrained_element));
  WR3: (‘ALL_ELEMENTARY_GEOMETRY’ IN
  TYPEOF(self\parametric_assembly_constraint.reference_element));
END_ENTITY;
(*
```

Attribute definitions:

Distance: Distance in three dimensions between **constrained_element** and **reference_element**.

Sense_check: This attribute is necessary only in the case that both constrained element and reference element are planes. If TRUE, plane normal vectors of the two planes are identical. If FALSE, they are opposite.

Side_check: This attribute is necessary if one or both of constrained element and reference element is a plane to distinguish if the element other than a plane exists in the front side of the plane or not. In case both elements are planes, this attribute implies the side information seen from the **reference_element**.

Centre_check: This attribute specifies if the distance should be measured to the geometry centre or to the boundary in case one of **constrained_element** and **reference_element** is a circle, or a cylindrical surface, or a spherical surface.

Near_points: Optionally used to specify desired solution when two or more constraint satisfaction is expected.

Formal propositions:

WR1: distance should be non negative.

WR2: constrained_element can be selected from **all_elementary_geometry** type.

WR3: reference_element can be selected from **all_elementary_geometry** type.

Informal propositions:

IP1: The preservation of the sense of constrained plane with regard to reference plane is regarded as part of the constraint since it distinguishes orientation of two component parts. This sense is not explicitly represented in the constraint entity, but will be determined by reference to the current result.

IP2: The preservation of the side of an element with regard to a plane is regarded as part of the constraint since it distinguishes position and orientation of two component parts. This side information is not explicitly represented in the constraint entity, but will be determined by reference to the current result.

IP3: If one of or both **constrained_element** and **reference_element** is a circle, **constrained_element** and **reference_element** should be coplanar.

IP4: line vs. **line** case: If two lines are coplanar, distance constraint implies that two lines are parallel. If two lines are not coplanar, specified distance is minimum distance between them, but two lines are not required to be parallel.

IP5: line vs. **plane**, or **cylindrical surface** vs. **plane** cases: A distance constraint applied between a line or a cylindrical surface and a plane implies that the line direction or the axis of cylindrical surface is perpendicular to the plane normal.

IP6: place vs. **plane** case: A distance constraint between two planes implies that the plane normal vectors are parallel.

IP7: cylindrical surface vs. **cylindrical surface**: Distance constraint to a cylindrical surface can be measured to its axis. In this case, a distance constraint between two cylindrical surfaces with coplanar axes will also imply that the axes are parallel. If two axes are not coplanar, specified distance is the minimum distance between them, but two axes are not required to be parallel. Distance can be measured to the nearest position on the surface, which implies minimum distance between them. If near_point is specified, constraint satisfaction the most close to the specified point is implied.

IP8: spherical surfaces, circles: A distance constraint to a spherical surface or to a circle is measured to its centre or to its boundary.

7.4.3 angle_constraint

This constraint specifies that a constrained element in one component part is constrained with respect to a reference element in another component part with specified angle. This constraint is defined between any pair of **directed_geometry**. The angle is measured between their forward directions and can take a value from zero to 180 degree inclusive.

An angle constraint between two elements does not require that those elements intersect. Two lines with an angle constraint will intersect only when they are coplanar.

EXPRESS specification:

```
*)
ENTITY angle_constraint
   SUBTYPE OF (parametric_assembly_constraint);
     angle: plane_angle_measure;
WHERE
   WR1: ((angle>=0) AND (angle<=180));
   WR2: (‘DIRECTED_GEOMETRY’ IN
   TYPEOF(self\parametric_assembly_constraint.constrained_element));
   WR3: (‘DIRECTED_GEOMETRY’ IN
   TYPEOF(self\parametric_assembly_constraint.reference_element));
END_ENTITY;
(*
```

Attribute definitions:

Angle: constrained angle between the **constrained_element** and the **reference_element**.

Formal propositions:

WR1: The angle value should be from zero to radians inclusive.

WR2: constrained_element should be **directed_geometry**.

WR3: reference_element should be **directed_geometry**.

7.4.4 Parallel_constraint

This constraint specifies that a constrained element in one component part is constrained parallel to a reference element in another component part. This constraint is valid between any pairs of

directed_geometry. In case both constrained element and reference element are planes, a BOOLEAN attribute is required to distinguish if plane normal directions are identical or opposite. As for line or cylindrical surface cases, this constraint does not force forward directions to be identical or opposite.

EXPRESS specification:

```
*)
ENTITY parallel_constraint
   SUBTYPE OF (parametric_assembly_constraint);
     sense_check : OPTIONAL BOOLEAN;
WHERE
   WR1: (‘DIRECTED_GEOMETRY’ IN
   TYPEOF(self\parametric_assembly_constraint.constrained_element));
   WR2: (‘DIRECTED_GEOMETRY’ IN
   TYPEOF(self\parametric_assembly_constraint.reference_element));
END_ENTITY;
(*
```

Attribute definitions:

Sense_check: This attribute is necessary only in the case that both constrained element and reference element are planes. If TRUE, plane normal vectors of the two planes are identical. If FALSE, they are opposite.

Formal propositions:

WR1: constrained_element should be **directed_geometry**.

WR2: reference_element should be **directed_geometry**

Informal proposition:

IP1: The preservation of the sense of constrained plane with regard to reference plane is regarded as part of the constraint since it distinguishes orientation of two component parts. This sense is not explicitly represented in the constraint entity, but will be determined by reference to the current result.

IP2: Specifying two entities to be parallel is not the same as putting on a zero angular dimension.

7.4.5 Perpendicular_constraint

This constraint specifies that a constrained element in one component part is constrained perpendicular to a reference element in another component part. This constraint is valid between any pair of **directed_geometry**. A line and an axis of cylindrical surface are not forced to intersect even if perpendicular constraint is applied between them.

EXPRESS specification:

```
*)
ENTITY perpendicular_constraint
   SUBTYPE OF (parametric_assembly_constraint);
WHERE
   WR1: ('DIRECTED_GEOMETRY' IN
   TYPEOF(self\parametric_assembly_constraint.constrained_element));
   WR2: ('DIRECTED_GEOMETRY' IN
   TYPEOF(self\parametric_assembly_constraint.reference_element));
END_ENTITY;
(*
```

Formal propositions:

WR1: constrained_element should be **directed_geometry**.

WR2: reference_element should be **directed_geometry**

Informal proposition:

IP1: Specifying two entities to be perpendicular does not force these entities to intersect.

7.4.6 coincident_constraint

This constraint specifies that two entities are identical when they are the same type entities, or an entity is **on** the other entity everywhere when they are different entity types. This constraint is valid for **directed_geometry** plus point. For example, if two lines are constrained coincident, they are forced to be identical. If a point and a plane are constrained coincident, the point should always be on the plane. It is not possible to apply a coincident constraint between a cylindrical surface and a plane. In case both constrained element and reference element are planes, a BOOLEAN attribute is required to distinguish if plane normal directions are identical or opposite.

EXPRESS specification:

```
*)
ENTITY coincident_constraint
   SUBTYPE OF (parametric_assembly_constraint);
     sense_check : OPTIONAL BOOLEAN;
     near_points: OPTIONAL SET[1:2] OF CARTESIAN_POINT;
WHERE
   WR1: ('DIRECTED_GEOMETRY' IN TYPEOF
   (self\parametric_assembly_constraint.constrained_element)) OR ('POINT'
   IN TYPEOF(self\parametric_assembly_constraint.constrained_element));
   WR2: ('DIRECTED_GEOMETRY' IN TYPEOF
   (self\parametric_assembly_constraint.reference_element)) OR ('POINT'
   IN TYPEOF(self\parametric_assembly_constraint.reference_element));
END_ENTITY;
(*
```

Attribute definitions:

Sense_check: This attribute is necessary only in the case that both constrained element and reference element are planes. If TRUE, plane normal vectors of the two planes are identical. If FALSE, they are opposite.

Near_points: Optionally used to specify desired solution when two or more constraint satisfaction is expected.

Formal propositions:

WR1: constrained_element should be a point or any entity of **directed_geometry** type.

WR2: reference_element should be a point or any entity of **directed_geometry** type.

Informal proposition:

IP1: The preservation of the sense of constrained plane with regard to reference plane is regarded as part of the constraint since it distinguishes orientation of two component parts. This sense is not explicitly represented in the constraint entity, but will be determined by reference to the current result.

7.4.7 cocentric_constraint

This constraint specifies that an axis of a constrained element in one component part is constrained identical to that of a reference element in another component part. This constraint is valid between two axial_geometry type entities. A point can also be made concentric with an axial geometry when it lies on the axis

EXPRESS specification:

```
*)
ENTITY cocentric_constraint
   SUBTYPE OF (parametric_assembly_constraint);
WHERE
   WR1: (‘AXIAL_GEOMETRY’ IN TYPEOF
   (self\parametric_assembly_constraint.constrained_element)) OR (‘POINT’ IN
   TYPEOF(self\parametric_assembly_constraint.constrained_element));
   WR2: (‘AXIAL_GEOMETRY’ IN TYPEOF
   (self\parametric_assembly_constraint.reference_element)) OR (‘POINT’ IN
   TYPEOF(self\parametric_assembly_constraint.reference_element));
   WR3: NOT((‘POINT’ IN TYPEOF(self\parametric_assembly_constraint.constrained_element))
   AND (‘POINT’ IN TYPEOF(self\parametric_assembly_constraint.reference_element)));
END_ENTITY;
(*
```

Formal propositions:
WR1: constrained_element should be a point or any entity of **axial_geometry_element** type.
WR2: reference_element should be a point or any entity of **axial_geometry_element** type.
WR3: It is not allowed to apply a **concentric_constraint** between two points.

7.4.8 tangent_constraint

This constraint specifies that a constrained element in one component part is constrained tangent with respect to a reference element in another component part. This constraint can be applied between a pair of elementary_curve_or_surface_geometry type entities. This constraint can not be applied between a pair of entities where both have zero curvature, for example a line and a plane.

EXPRESS specification:

```
*)
ENTITY tangent_constraint
   SUBTYPE OF (parametric_assembly_constraint);
   near_points: OPTIONAL SET[1:2] OF CARTESIAN_POINT;
WHERE
   WR1: (‘ELEMENTARY_CURVE_OR_SURFACE_GEOMETRY’ IN TYPEOF
   (self\parametric_assembly_constraint.constrained_element));
   WR2: (‘ELEMENTARY_CURVE_OR_SURFACE_GEOMETRY’ IN TYPEOF
   (self\parametric_assembly_constraint.reference_element));
   WR3: NOT(((‘LINE’ IN TYPEOF(self\parametric_assembly_constraint.constrained_element))
   AND (‘LINE’ IN TYPEOF (self\parametric_assembly_constraint.reference_element))) OR
   ((‘LINE’ IN TYPEOF (self\parametric_assembly_constraint.constrained_element)) AND
   (‘PLANE’ IN TYPEOF (self\parametric_assembly_constraint.reference_element))) OR
   ((‘PLANE’ IN TYPEOF (self\parametric_assembly_constraint.constrained_element)) AND
   (‘LINE’ IN TYPEOF (self\parametric_assembly_constraint.reference_element)))OR ((‘PLANE’
   IN TYPEOF(self\parametric_assembly_constraint.constrained_element)) AND (‘PLANE’ IN
   TYPEOF(self\parametric_assembly_constraint.reference_element))));
END_ENTITY;
(*
```

Attribute definitions:

Near_points: Optionally used to specify desired solution when two or more constraint satisfaction is expected.

Formal propositions:

WR1: constrained_element should be selected from **elementary_curve_or_surface_geometry** type.

WR2: reference_element should be selected from **elementary_curve_or_surface_geometry** type.

WR3: It is not allowed to specify zero curvature entity for both **constrained_element** and **reference_element** such as a line and a line, a line and a plane, and a plane and a plane.

Informal propositions:

IP1: curve vs. curve case: Tangent vectors of the two curves are parallel at the point of contact.

IP2: curve vs. surface: Tangent vector of the curve is perpendicular to the surface normal vector at the point of contact.

IP3: surface vs. surface: Surface normal vectors are parallel at the point of contact
IP4: If one of or both **constrained_element** and **reference_element** is a circle, **constrained_element** and **reference_element** should be coplanar.

EXPRESS specification:

```
*)
END_SCHEMA; -- parametric_assembly_constraint
(*
```

8 Schema fixed_component_constraint

8.1 Introduction

The following EXPRESS declaration begins the **fixed_component_constraint_schema** and identifies the necessary external references.

EXPRESS specification:

```
*)
SCHEMA fixed_component_constraint;
   REFERENCE FROM assembly_model_schema
       (mechanical_part_definition);
(*
```

8.2 Fundamental concepts and assumptions

This schema provides the resources for the specification of fixed component constraint which is used in conjunction with the **parametric_assembly_constraint** schema. It is provided to enable constrained assembly configuration to be anchored in space. Without its use, assembly constraint equations may have infinitely many solutions due to the existence of translational and rotational degrees of freedom. The implication of the constraint is that specified component part may not be edited, either by the user or by a constraint solver.

8.3 fixed_component_constraint entity definitions

This clause contains ENTITY definitions of the **fixed_component_constraint schema.**

8.3.1 fixed_component_constraint

This entity is provided to specify a component instance which should be fixed in parametric assembly situation.

<u>EXPRESS specification</u>:

```
*)
ENTITY fixed_component_constraint;
  fixed_component_instance: mechanical_part_definition;
END_ENTITY;
(*
```

<u>Attribute definitions</u>**:**

Fixed_component_instance: Instance of a component part which should be fixed in parametric assembly situation.

<u>EXPRESS specification</u>:

```
*)
END_SCHEMA; -- fixed_component_constraint
(*
```

Annex A
(informative)
EXPRESS-G diagram